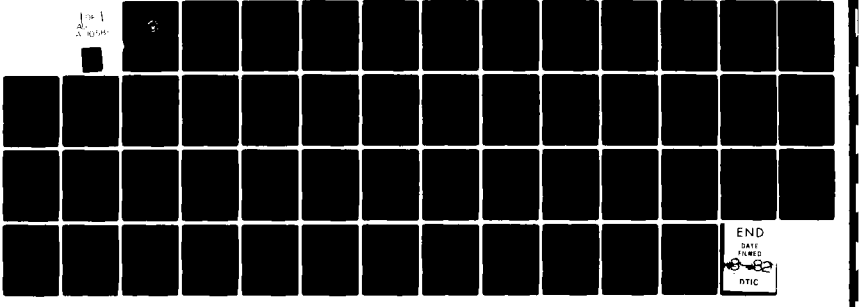
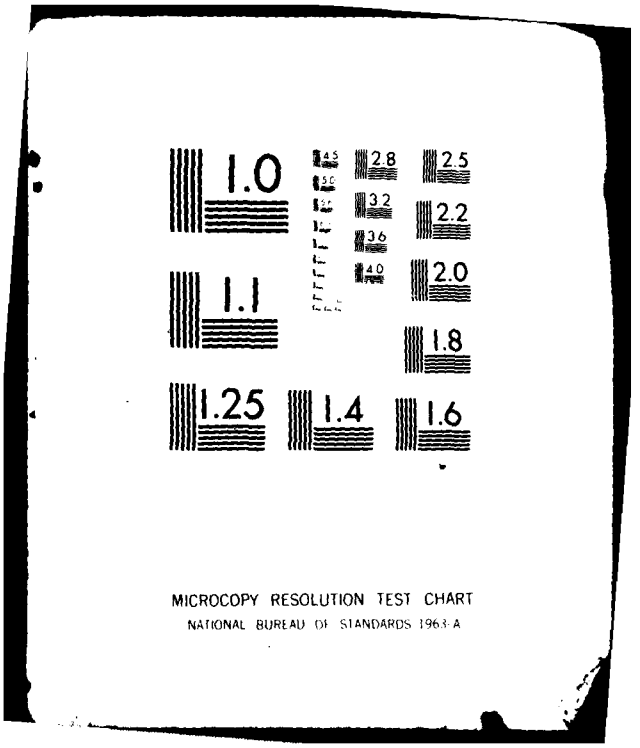
AD-A110 581 NAVAL POSTGRADUATE SCHOOL MONTEREY CA F/G 9/2
STATISTICS PROGRAMS FOR THE APPLE II PLUS MICROCOMPUTER.(U)
MAR 81 J D MORGESON

UNCLASSIFIED NL

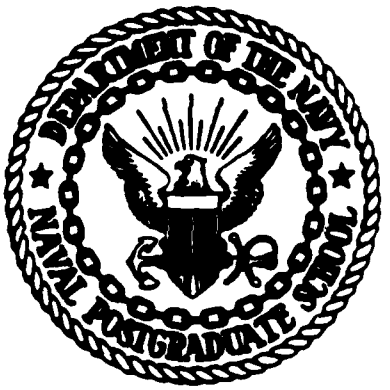


END
DATE FILMED
3-82
DTIC


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963-A

AD A110581

NAVAL POSTGRADUATE SCHOOL
Monterey, California



THESIS

STATISTICS PROGRAMS FOR THE
APPLE II PLUS MICROCOMPUTER

by

James Darrell Morgeson

March 1981

Thesis Advisor D. R. Barr

Approved for public release, distribution unlimited

FILE COPY

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER	2. GOVT ACCESSION NO. AD-A110581 / 3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Statistics Programs for the Apple II Plus Microcomputer	5. TYPE OF REPORT & PERIOD COVERED Master's Thesis; March 1981
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) James Darrell Morgeson	8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940	12. REPORT DATE March 1981
	13. NUMBER OF PAGES 51
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) Naval Postgraduate School Monterey, California 93940	15. SECURITY CLASS. (of this report) Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Apple II Plus
Microcomputer
Confidence Interval
Hypothesis Tests
Distribution Approximations

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This paper presents a statistical software package developed for use on the Apple II Plus microcomputer, modified with the Apple Pascal language card. The program addresses the following: determination of confidence intervals for single and bivariate populations; hypothesis testing for one and two paramenters; computation of cumulative distribution values for the Normal, Student's T,..

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

Chi-square, F, Binomial and Poisson distributions; computation of quantile values for the Normal, Student's T, Chi-square and F distributions. The program also has the capability to store, retrieve, and modify data for use with the statistical procedures. The program was written in UCSD Pascal, which because of its portability indicates that little or no modification would be required to use it with other computers which are UCSD Pascal compatible. In addition, because of Pascal's block structure, the program can be easily modified or enhanced to include other statistical procedures which are of interest to the user.

Accession For
NTIS GRA&I
DTIC TAB
Unannounced
Justification
By
Distribution
Availability
Dist
A

DTIC
COPY
INSPECTED
2

Approved for public release; distribution unlimited.

Statistics Programs for the Apple II Plus Microcomputer

by

James Darrell Morgeson
Major, United States Army
B. S., United States Military Academy, 1971

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN OPERATIONS RESEARCH

from the

NAVAL POSTGRADUATE SCHOOL
MARCH1981

Author James D. Morgeson

Approved by: _____
Thesis Advisor

_____
Second Reader

_____
Chairman, Department of Operations Research

W M Woods
Dean of Information and Policy Sciences

ABSTRACT

This paper presents a statistical software package developed for use on the Apple II Plus microcomputer, modified with the Apple Pascal language card. The program addresses the following: determination of confidence intervals for single and bivariate populations; hypothesis testing for one and two parameters; computation of cumulative distribution values for the Normal, Student's T, Chi-square, F, Binomial and Poisson distributions; computation of quantile values for the Normal, Student's T, Chi-square and F distributions. The program also has the capability to store, retrieve, and modify data for use with the statistical procedures. The program was written in UCSD Pascal, which because of its portability indicates that little or no modification would be required to use it with other computers which are UCSD Pascal compatible. In addition, because of Pascal's block structure, the program can be easily modified or enhanced to include other statistical procedures which are of interest to the user.

TABLE OF CONTENTS

I. INTRODUCTION- 6

II. BACKGROUND- 8

A. DESCRIPTION OF THE MICROCOMPUTER SYSTEM - - - 8

B. SOFTWARE DEVELOPMENT- - - - - - - - - - - - - 8

C. PORTABILITY - - - - - - - - - - - - - - - - - -10

D. EASILY RETRIEVABLE INFORMATION FILES- - - - -10

E. SIMPLE USER INTERFACE - - - - - - - - - - - -11

III. ALGORITHMS- - - - - - - - - - - - - - - - - - - -13

A. CALCULATION OF VARIANCE - - - - - - - - - - -13

B. DISTRIBUTIONS AND INVERSES- - - - - - - - - -14

C. POISSON AND BINOMIAL DISTRIBUTIONS- - - - - -15

D. STATISTICAL ALGORITHMS- - - - - - - - - - - -16

IV. DESCRIPTION OF PACKAGE- - - - - - - - - - - - - -17

A. THE DATA ENTRY MODULE - - - - - - - - - - - -17

B. DISTRIBUTIONS AND QUANTILES MODULES - - - - -25

C. CONFIDENCE INTERVALS AND HYPOTHESIS TESTING MODULES -27

V. CONCLUSIONS - - - - - - - - - - - - - - - - - - -31

APPENDIX A - STATISTICAL ALGORITHMS - - - - - - - - - -33

APPENDIX B - STRING CONVERSION ALGORITHM- - - - - - - -47

LIST OF REFERENCES- - - - - - - - - - - - - - - - - - -49

BIBLIOGRAPHY- -50

INITIAL DISTRIBUTION LIST - - - - - - - - - - - - - - -51

I. INTRODUCTION

For the operations analyst or for that matter anyone who utilizes the methodology and problem solving approach of the operations analyst, some form of computational device is a necessity. Cost, rigid interface requirements, and a number of other factors have in the past frustrated and stifled the analyst in bringing to bear the necessary computational power on his problem. In the late 1970's, however, Texas Instruments introduced a significant amount of computing power packaged in the programmable TI-59 calculator. Revolutionary is perhaps a bit too strong as a description of the impact that this and like devices have made in the operations research community, but certainly most analysts would agree that the amount of computing power that can now be held in one's hand and used to solve problems is indeed remarkable.

In spite of this impact, however, the hand-held calculator's contribution might even now be waning and yielding to a more spectacular capability found in the microcomputer. The growth in the capability of these devices since the beginning of the industry in 1971 has been phenomenal. Mastrakas details this growth and gives a glimpse of the possible direction that the 1980's will see in this industry [Ref. 1]. One factor accounting for the growth in microcomputer technology and one that serves to insure its future is the intense competition that pervades the industry. When one thinks of hand-held calculators, he thinks of Texas Instruments or perhaps Hewlett-Packard. When one thinks of microcomputers, he may think of Apple, Exidy, North Star, PET, TRS-80, or a number of other devices with like capability.

A typical microcomputer package costing between $2,000 and $2,500 might consist of the following: the computer (about the size of a small portable typewriter) with 64 K bytes of usable random access memory, two floppy-disk drives (5 1/4 inch diameter) for program storage, and a black and white monitor for output display. A variety of programming languages are also available including the University of California, San Diego (UCSD) Pascal, BASIC, FORTRAN, PILOT, etc. Some of the microcomputers also have a graphics capability allowing the user to visualize mathematical forms, plot graphs, and plot data observations.

The American populace is being conditioned through current periodicals and news features to expect an increasingly important role for the microcomputer in everyday life — from grocery shopping to environmental control for the home. The December 1, 1980 issue of "U. S. News and World Report" predicts that eighty percent of the United States' households will have a microcomputer by 1990. In view of the prospective proliferation of these devices and also the significant computing power that now exists, the operations research analyst can ill afford not to begin to exploit the capabilities of microcomputers. Indeed, the hardware capabilities have grown and are growing so rapidly that today good compatible software to support these impressive capabilities is seriously lacking in specialized fields such as operations research. It is this software deficiency that this thesis addresses.

II. BACKGROUND

The operations analyst can, on occasion, be required to establish confidence intervals or test hypotheses about an unknown parameter from a known or assumed population. The computer program written as a part of this thesis allows the analyst to quickly and easily accomplish these tasks when observations are from Normal, Exponential or Bernoulli populations.

A. DESCRIPTION OF THE MICROCOMPUTER SYSTEM

The software development was done on an Apple II Plus microcomputer with two floppy-disk drives (5 1/4 inch diameter) as add-on peripherals. The system is equipped with a language system giving the capability to use the University of California, San Diego (UCSD) Pascal programming language as well as the BASIC language which comes resident with the computer. The standard Apple computer has a forty column output display which makes it compatible for use with a standard television set. This capability can be enhanced to an eighty column display with an additional peripheral device, provided a monitor is used for display in lieu of a television. The output format for the program is written for an eighty column display device; however, using the special built-in features of the Apple, the format can easily be made to display split screen on a system not equipped with an expanded display peripheral.

B. SOFTWARE DEVELOPMENT

Ling and Muller give several considerations which should be observed in the development of software for statistical analysis [Ref. 2, Ref. 3]. Among these considerations are the following:

1. Choice of Programming Language

The Pascal programming language used in this software development effort has a number of features which make it particularly attractive for use with microcomputers. First, Pascal is a very concise language. The compiler is small and compact and fits easily within the available off-line storage space of the floppy-disk. Second, Pascal is a high-level, general-purpose language [Ref. 4]. The language was originally introduced in 1971, which is recent in comparison to most high-level languages. It was intended to be used in teaching new programmers good techniques and style. The Pascal language fully exploits the fundamental concepts of structured programming, which is a technique used by many professional programmers to write large complex computer programs [Ref. 4]. The use of these techniques facilitates developing programs in a modular fashion (i. e., break the overall package into logical sub-packages and proceed to program, debug, and validate each sub-package individually). The final step in the process is to combine the sub-packages to form the overall package. Using the attendant statistical package as an example, six sub-packages or modules comprise the overall package. Each module is independent of the others and can stand alone when compiled with the main program. Third, the Pascal language performs arithmetic computations significantly faster than the BASIC language. Pascal is often implemented as a "pseudo" interpreted language meaning that the text versions of programs are first compiled into a code file. It is during the compilation phase that syntax errors are detected by the compiler and brought to the programmer's attention. This code file is interpreted and executed during the execution phase of the program. Most processors can and typically do interpret a Pascal code file significantly

faster than a corresponding BASIC program which performs the same computations [Ref. 4].

2. Computational Efficiency

In spite of their impressive capabilities, microcomputers are decidedly inferior to larger computers in the two key areas of computational speed and accuracy. Pascal, as implemented on the Apple, will only perform computations using six decimal places of accuracy [Ref. 5] and displays only five places past the decimal. Accordingly, algorithms which are used and work very well on larger computers might have no chance of producing the same results on microcomputers simply because the execution time is excessive, or they require double precision arithmetic.

C. PORTABILITY

A portable program is one that can be run on a number of different computer systems [Ref. 2]. Programs written in the Pascal language are valuable in this regard in that they may be run on a variety of microcomputers without alteration. Some of the machines for which this is true are computers with the following microprocessors: 8080, 8085, Z80, 6502(Apple), 6800, and 9900. Portability is a very strong asset of the Pascal language even though it is accomplished at the expense of reduced computational speed. The compiled code version of programs is called "p-code". Each machine has a special interpreter program which takes the "p-code" and converts it to a form compatible with the existing host microprocessor.

D. EASILY RETRIEVABLE INFORMATION FILES

Ling suggests the use of help files to allow the user to make efficient use of a program [Ref. 2]. The number of information files which are available and their content is clearly a matter of judgment based on assumptions concerning the knowledge of the user population about the procedures

used in the program. Anticipating that most of the users of the accompanying statistical package will be familiar with the basic concepts of the procedures themselves, only one help file which pertains to data entry requirements is included.

E. SIMPLE USER INTERFACE

The user is called upon to make numerical entries throughout the program. Because Pascal is a strongly typed language, a variable of type integer cannot be assigned floating point values. Indeed, this requirement is so strict that if the program expects the user to enter an integer and he accidently enters a number in decimal point notation, the program will abort and cause the entire system to re-initialize. Situations such as this are, of course, undesirable. Moreover, all programs written with the intent of establishing a dialogue between the program and the user should be as trouble-free as possible for the user and minimize as much as possible the user's chances of committing a fatal error when responding to program prompts or entering data. In general, schemes to accomplish this are costly in terms of computational efficiency and programming steps; however, user convenience is almost always worth the costs. The problem with data entry or numerical entry from the keyboard is addressed in this statistical package by making all numerical entries using a string and converting the string to the numerical value it represents. A string is a variable type which is a linear array of characters. For example, given a string variable called 'S', which is assigned a value of '235', the character in array position 'S[1]' is '2', 'S[2]' is '3', and 'S[3]' is '5'. Since the characters '0' through '9' have corresponding numerical American Standard Code for Information Interchange (ASCII) values of forty-eight

through fifty-seven, conversion is accomplished by subtracting forty-eight from the character's ASCII value. A Pascal procedure which converts strings to numbers is shown in Appendix B.

The advantages in using a scheme such as this for all numerical entries are two-fold. First, the user can take advantage of the direct cursor addressing available on many microcomputer video displays to correct a data entry prior to entering it into computer memory. This is not possible when the program expects real numbers as input. Second, the program segment which converts the string may return either an integer or a real number, whichever is required by the program. This avoids having the user concerned with the typing requirements demanded by the Pascal language.

III. ALGORITHMS

Algorithms used for statistical computations on microcomputers should be selected with the goal of providing the best accuracy achievable at the minimum computation time. Given the limitation of only six decimal places of accuracy on the Apple microcomputer, many algorithms requiring more precision in their computations must be rejected. The algorithms remaining as candidates must be carefully screened to insure that their required computations are reasonable from a time standpoint and that they exercise fully the accuracy capability of the machine.

A. CALCULATION OF VARIANCE

To illustrate the complications stemming from reduced accuracy, consider the following example given by Forsythe [Ref. 6].

Find the variance of the following set of numbers:
48499, 48503, 48500, 48498, 48500
The definition for the variance is

$$S^2 = \frac{1}{N-1} \sum_{i=1}^{N} (X_i - \bar{X})^2$$

where S^2 = variance
N = number of observations
$\bar{X}$ = sample mean.

This formula can be expanded to the following form:

$$S^2 = \frac{1}{N-1} \sum_{i=1}^{N} X_i^2 - N\bar{X}^2$$

Use of this formula in the Apple, however, would yield a variance of zero, simply because the required accuracy is not available for computations. The answer of zero is, of course, incorrect. Forsythe in his article on statistical computing offers the following alternative algorithm to compute the variance.

```
SUM := 0.0;
S2 := 0.0;
READ (N);
FOR I := 1 TO N DO
BEGIN
   READ (X);
   DEVIATION := X - SUM;
   SUM := SUM + DEVIATION/I;
   S2 := S2 + DEVIATION * (X - SUM);
END;
S2 := S2/(N - 1);
```

This algorithm produces the correct answer, "variance = 3.5". It also illustrates how many of the limitations of the microcomputer can be overcome through careful selection of algorithms.

B. DISTRIBUTIONS AND INVERSES

The algorithms used to compute probability distributions and inverses and the source of each are listed in Appendix A. When compared to the tabular values listed in Dixon and Massey [Ref. 7], the algorithms are accurate to at least three decimal places in probability with the exception of the F distribution. The F distribution is accurate to three decimal places in probability in almost all cases; however, some values may differ from the listed tabular values by as much as .002 in probability. Although the F quantiles may differ slightly from the listed tabular values, the probabilities corresponding to the values given by the algorithm are accurate to three decimal places. All of the algorithms produce results within three seconds except the T distribution. When computing for very large degrees of freedom for the T distribution, computation time is a function of the degrees of freedom. Typically 1000 degrees

of freedom takes approximately eight seconds.

Often, increased accuracy can only be gained through iterative techniques. This is the case with many of the probability distributions and their inverses, the T distribution being an example. In the algorithms shown in Appendix A much of the excessive computation time has been alleviated by combining two or more algorithms. For example, in the Chi-square distribution, the F distribution and the inverse F distribution, one algorithm is used for small degrees of freedom and another for large degrees of freedom. Good algorithms for these distributions exist for large degrees of freedom which are not based on iterative techniques and hence are computationally fast. However, for small degrees of freedom, their accuracy falls off rapidly. Conversely, the algorithms using iterative techniques are very accurate at all ranges, but slow for the larger degrees of freedom since the number of iterations required is proportional to degrees of freedom. The break between small and large degrees of freedom is purely subjective, based on choosing the best combination of speed and accuracy. Selection of the algorithms themselves was likewise based on the best combination of speed and accuracy.

C. POISSON AND BINOMIAL DISTRIBUTIONS

Large mainframes have the capability of computing the Poisson and Binomial Distributions directly from their definitions.

POISSON:

If the random variable X is distributed Poisson with parameter $\{\lambda\}$

$$\text{then } P[X <= k] = \sum_{j=0}^{k} \frac{\lambda^j e^{-\lambda}}{j!} .$$

BINOMIAL:

If the random variable X is distributed Binomial with parameters $\{n, p\}$,

then $P[X <= k] = \sum_{j=0}^{k} \binom{n}{j} p^j (1-p)^{n-j}$.

Both of these expressions contain factorials and summations; consequently for large values of 'k' in the Poisson and 'n' and 'k' in the Binomial, execution time might be excessive, or the intermediate values in computation might exceed '10^{38}' (the maximum number capable of being represented on the Apple). A better solution for the calculation of these probabilites is to use the Chi-square identity for the Poisson and the F identity for the Binomial [Ref. 8, Ref. 9].

POISSON:

Given $X \sim$ Poisson

then $Pr[X <= k] = 1 - \chi^2_{(2k+2)}(2\lambda)$

where 2k + 2 are the degrees of freedom of the Chi-square variate.

BINOMIAL:

Given $X \sim$ Binomial $\{n, p\}$

then $P[X <= k] = P\left[F_{(2n-2k,\ 2k+2)} <= \frac{1-p}{p} \cdot \frac{k+1}{n-k}\right]$

where 2n-2k equals the degrees of freedom for the numerator and 2k+2 equals the degrees of freedom for the denominator.

When k = n, the probability is 1.0.

D. STATISTICAL ALGORITHMS

Derivation of the algorithms used in finding confidence intervals and hypothesis testing is shown in Storer [Ref. 10].

IV. DESCRIPTION OF PACKAGE

The options available to the user are shown in the block diagram (Figure 1). When the program is executed, the user begins in the "outer level". To proceed he must select from one of the seven options which are shown on a "menu" (Figure 2).

A. THE DATA ENTRY MODULE

Selecting option '6' from the outer level menu (Figure 2) will cause the data entry module menu (Figure 3) to appear on the screen.

1. General

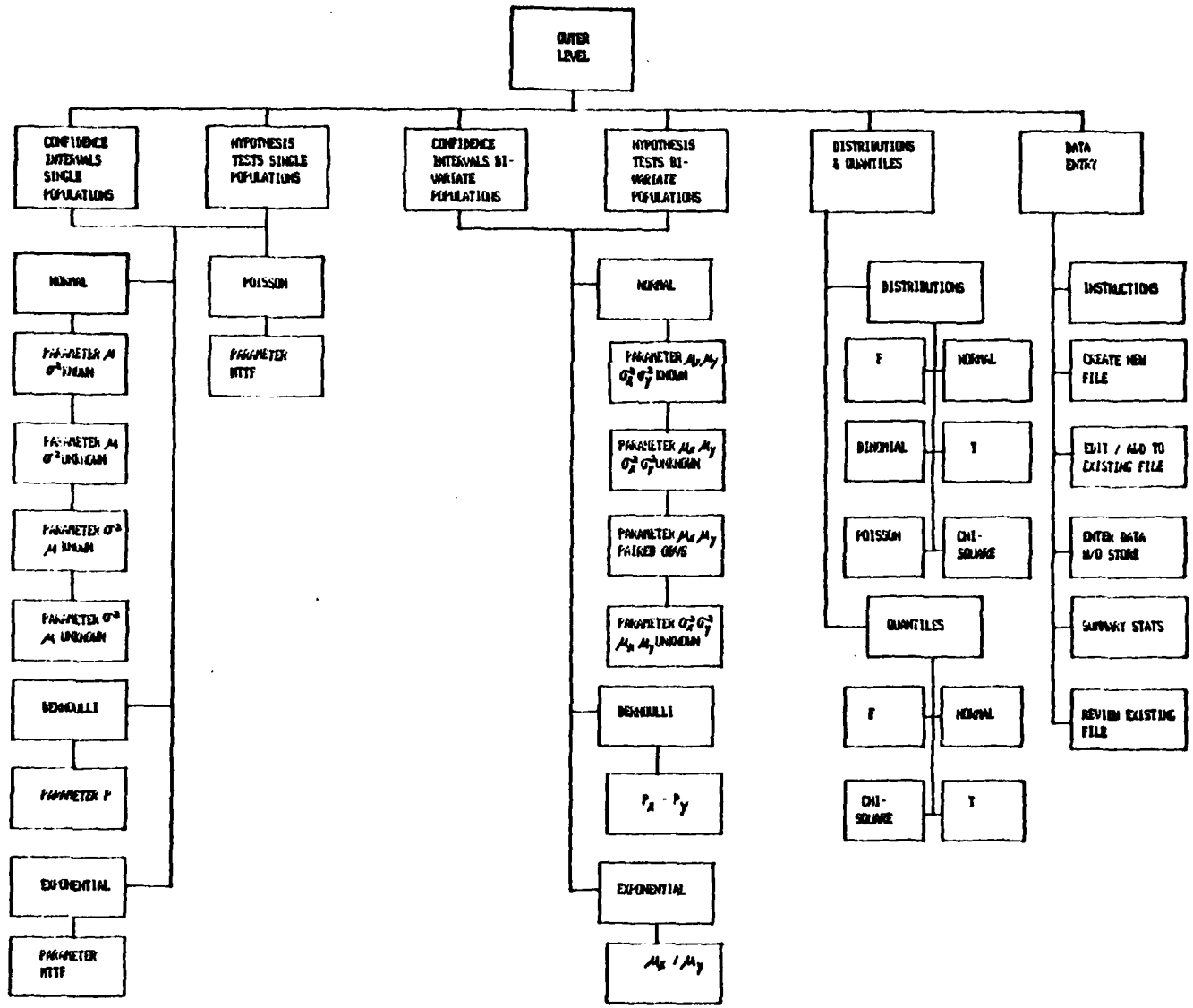
The data entry requirements are intended to be as trouble-free as possible for the user. The user is prompted for data input by the following line:

Record N -->

'N' is an integer sequentially updated by the program when the 'return' key is pressed. Following the arrow, the user inputs as many data values as he wishes with entries separated by one or more spaces. The only restriction is that he should not exceed the length of the display line. The nomenclature, record, indicates only a logical or convenient grouping of data from the user's point of view.

Prior to entering any data, the user is asked whether or not the observations he is entering are paired. Since two of the statistical procedures are predicated on paired data, answering 'yes' to this question will cause summary statistics to be computed on 'X,Y' pairs. These summary statistics are only good for use with the procedures requiring paired observations. All data entered

Figure 1. Block Diagram



1) Hypothesis Testing One Parameter
2) Hypothesis Testing Two Parameters
3) Confidence Intervals Single Population
4) Confidence Intervals Bivariate Populations
5) Distributions and Inverses
6) Data Entry
Q) uit

Figure 2. The Outer Level Menu

1) Instructions
2) Create a new data file
3) Correct/Add to existing datafile
4) Enter data without storing
5) Review existing datafile
6) Review summary statistics of existing file
Q) uit and return to outer level

Figure 3. The Data Entry Module Menu

must be in 'X,Y' pairs with the 'X' observation listed first. All other statistical procedures and their attendant data entry requirements assume that all of the observations come from a single population. These general requirements are available to the user when he selects option '1' from the data entry menu (Figure 3).

2. Create A New Data File

Selecting option '2' (Figure 3) will prompt the user to specify a file name for the data observations. After responding with a file name, the operating system establishes a directory entry at the beginning of the largest unused block of space containing at least fifteen blocks on the specified disk. Because the filing system in the University of California, San Diego (UCSD) Pascal language is random access, each data file entered will have allocated fifteen blocks of space to insure that there is enough room to extend the file if the user desires to do so at some later time.

The floppy-disk (5 1/4 inch diameter) used by the Apple System provides a storage space of 280 blocks. This results in a capability to store seventeen data files on each disk. Each data file can contain a maximum of ninety records. Since each record can contain as many observations as the user desires provided that it does not exceed the length of the display line, a reasonable planning figure is eight data observations per record. This results in an upper limit of approximately 720 data entries per data file.

Data entry for a new file begins with 'Record 1'. The user's only concerns when entering data should be to separate each observation by one or more spaces, to not exceed the length of the display line, and to enter 'X,Y' pairs if he has previously indicated paired observations.

Since the data is entered as a string variable and then converted, it is possible to correct any entry prior to going to the next record by simply using the 'back arrow' key to move to the place at which the correction is to be made. The legal symbols which may be used in data entry are the digits '0' through '9', the decimal point, the comma, the plus (+) and minus (-) signs, and the 'E' for scientific notation. Commas are ignored by the procedure which converts the strings and are included only as a convenience for the user.

Pressing the 'return' key at the end of a record terminates that record and prompts the user to input the next record in sequence. If the user inadvertantly enters an illegal character while entering data, the program will advise him of this, indicate what the character was, and prompt him to reenter the record.

Once the user has entered all of his data, he must press the 'escape' key and the 'return' key following the last data entry. This will close the data file. When the file is closed, the following summary statistics are computed on the observations in the file:

a. The Sample Mean
b. The Sample Standard Deviation (N-1)
c. The Sum of the Observations
d. The Sum of Squares of the Observations
e. The Number of Observations

For paired data, the same statistics are computed; however, they are computed on the differences of the 'X,Y' pairs. Hence, the number of observations for a data file of paired observations is exactly half of the total number of data entries.

The summary statistics for each data file are kept in a separate file that is initially established on the same disk as newly created data files. The file of summary statistics requires one block of disk storage space. It is differentiated from the original file of observations by an 'S' concatenated to the original file name.

3. Correct/Add to Existing Data File

Selecting option '3' (Figure 3) while in the data entry module will prompt the user to specify a file name. Once the data file is retrieved, the monitor will show the name of the file, the number of records in the file, and the number of observations. Immediately after this information is the following prompt:

```
Enter Record Number -->
```

Selecting any record number between '1' and the total number of records in the file will cause the retrieval of that record and will display as follows:

```
Old Record:
Record 2 -->  1 2 3 4 5
You may replace the complete record or
press <RTN> for no changes.
Record 2 -->
```

Pressing 'return' leaves the existing record unchanged and prompts the user to enter another record number. If corrections are to be made, the user must enter the complete new record opposite the lower prompt and press 'return'. The updated record will display as follows:

```
Record 2 --> 1 2 3 3 5
Press <RTN> if OK, <ESC> if not.
```

Pressing 'return' completes the update and prompts for a new record. Pressing 'escape' produces the following display:

```
Enter corrected data and <RTN>
Record N -->
```

The user then may retype the line. This sequence may be repeated until the record appears as the user wants. Entire records are erased or deleted by typing a space to produce a blank line. If the user desires only to update an existing data file and not extend it, he types the number, '-1', in response to the prompt for record number, which closes the updated file.

If the user selects a record number that is greater than the number of records in the file, he reenters the data entry phase beginning with the record number immediately following the last existing record. For example, if there are thirty-seven existing records in the file, selecting any number greater than thirty-seven will produce the following display:

```
Enter new records.
Enter <ESC> as the last entry and <RTN> to terminate input.
Record 38 -->
```

From this point, the user proceeds exactly as if he were in the data entry phase and terminates by pressing the 'escape' key immediately following the last entry.

4. Enter Data Without Storing

This option allows the user to compute the mean, standard deviation, sum of observations, sum of squares of observations, and number of observations. The format for entering data is exactly the same as previously discussed for creating a new file. As indicated, data is not stored under this option; hence, once a record is terminated, the entries for that record cannot be recovered nor changed. This option provides an expedient way to determine the summary statistics of a group of observations.

5. Review an Existing Data File

Selecting this option allows the user to quickly review any or all of an existing data file in blocks of ten records at a time. The user is prompted to enter the file name of an existing file. The program retrieves that file and displays the first ten records. Pressing 'return' at this point causes the next ten records to be displayed, etc. Pressing the 'escape' key at the end of any display, returns the user to the menu for the data entry module (Figure 3).

6. Review Summary Statistics of an Existing File

All files that are stored on disk have an associated summary statistics file that is created by the program when a newly created or updated data file is closed. This file contains the following information:

a. Mean
b. Standard Deviation
c. Sum of Observations
d. Sum of Squares
e. Number of Observations

The summary statistics file only is called when specifying a data file to be used in the other modules containing the statistical procedures. Because of this, it is not necessary to have the original data file on-line when performing the statistical procedures; only the summary statistics file is required. When using the filing system resident in the Apple Pascal language system to obtain directory listings of various disks, the summary statistics file is distinguished by a concatenated 'S' on the end of the original file name. For example, if the original file name was STAT:DATA1, then the corresponding summary statistics file is named STAT:DATA1S. Since the length of all summary statistics files is one block, it is possible to access a maximum of 274 files of summary statistics on any

one disk of storage.

7. Quit

Pressing the 'Q' key will return the user to the outer level (Figure 2).

B. DISTRIBUTIONS AND QUANTILES MODULE

Selection of any of the options from the distributions and quantiles module menu (Figure 4) will produce further prompts which require the user to enter the necessary information concerning values of the random variable, degrees of freedom, and probabilities as appropriate. After each computation the following prompt appears:

C)ontinue or Q)uit

Pressing the 'C' key will allow the programmer to continue calculation in the previously selected distribution or quantile. Pressing the 'Q' key will return the user to the menu for this module (Figure 4).

The Pascal language system allows the user to develop his own specialized libraries of often used subroutines for general-purpose or special-purpose computations. It is in such a library that the algorithms for the distributions and quantiles are kept. Using a special library has two major advantages. First, when the user is developing the main program, he is not penalized by extra compilation time for any of the routines in the library. The code in the library is linked by a separate process [Ref. 4]. Second, the library can be used by other programs which require the use of the algorithms contained therein. The algorithms for the distributions and quantiles fit logically in a library since it is likely that other statistical packages will require their use. The Apple reference manual explains the procedure used to establish new libraries [Ref. 11].

1) Normal Distribution
2) T Distribution
3) Chi-Square Distribution
4) F Distribution
5) Binomial Distribution
6) Poisson Distribution

A) Normal Quantiles
B) T Quantiles
C) Chi-Square Quantiles
D) F Quantiles

Q) uit and return to outer level

Figure 4. The Distributions and Quantiles Module Menu

POPULATION ASSUMPTIONS

	Population	Assumptions	Parameter
1)	Normal	u?,sigma-sqr known	u
2)	Normal	u & sigma-sqr ?	u
3)	Normal	u & sigma-sqr ?	sigma-sq
4)	Normal	u known, sigma-sqr ?	sigma-sq
5)	Bernoulli		p
6)	Exponential		MTTF
Q)	uit and return to outer level		

u ==> mean
sigma-sq ==> variance

Figure 5. Confidence Intervals Single Population Menu

C. CONFIDENCE INTERVALS AND HYPOTHESIS TESTING MODULES

Selecting either the confidence intervals or hypothesis testing options will cause the menu for that module to be displayed. All of these menus contain information similar to that shown in figure 5 for confidence intervals.

The parameters about which an interval is to be computed or a hypothesis is to be made are listed in the right hand column of the menu. The distribution from which the observations came and assumptions about the populations are listed in columns one and two, respectively.

1. Data Requirements

When one of the options is selected that does not involve Bernoulli or Poisson populations the following display appears:

1) Use existing data file
2) Enter data and store it
3) Enter data w/o storing
4) Use summary statistics

Because of the nature of the observations, any tests or intervals involving Bernoulli or Poisson observations are entered using summary statistics; hence, for these cases, this display is skipped.

a. Use existing Data File

The user is prompted for the name of the data file. When the file name is entered, the program retrieves the summary statistics file associated with that file from disk. In the case of bivariate populations, two file names are needed, the first containing the 'X' observation and the second containing the 'Y' observation. Paired observations, as noted previously, are entered in one file.

b. Enter Data and Store it

Selecting this option will display a short message to the user informing him that all data storage must be accomplished from the data entry module (Figure 3). The

user has the option at this point to enter the data without storing it or to return to the outer level (Figure 2) and select the data entry module.

c. Enter Data W/O Storing It

Data is entered in the same format as previously discussed in the section pertaining to the data entry module. No permanent disk record is made of the entries. Hence, once each record of observations is terminated, there is no way to retrieve it to make corrections. Where bivariate populations are required, the user is prompted to enter all of the 'X' observations first and all of the 'Y' observations second. For paired observations all of the 'X,Y' pairs are entered as one population.

d. Use Summary Statistics

On all other tests or intervals, use of summary statistics is optional except as previously mentioned for data from Bernoulli or Poisson populations. At each prompt, the user is asked only for the necessary information to perform the statistical procedure he has selected. The distinction is made in each prompt whether or not the statistics required are the sample parameter values (sample mean, sample standard deviation) or the true parameter values (true mean, true standard deviation).

2. Confidence Intervals

When computing confidence intervals, the user must supply the additional information concerning the desired level of confidence and the type of interval (two-sided, one-sided upper, one-sided lower). Typically, computation of the desired interval takes one or two seconds and is displayed as follows:

```
 95.00 Percent confidence intervals for u
Sample Mean =  3.000
Standard deviation =  1.581
Upper =  4.964
Lower =  1.036
Another interval using same data, Y)es N)o -->
```

Following the display, the user is asked whether or not he desires to compute another interval using the same data. If he responds 'yes', he may then vary the confidence level and/or the type of interval without having to again specify the data base. Answering 'no' will return the user to the menu for the module in which computations are currently being performed.

3. Hypothesis Testing

Hypothesis tests require the user to specify the null hypothesis. Typically, the hypotheses involve 'equal to', 'less than or equal to', or 'greater than or equal to' comparisons and are displayed for the user in a form similar to the one below:

```
1) u  = u[0]
2) u <= u[0]
3) u >= u[0]
```

The symbol [0] represents the null hypothesis value. The user enters this value if required by the test. An example display following computation is as follows:

```
HYPOTHESIS:  u = u[0]
Sample mean =     3.000
u[0] =  2.500
The P-value is 0.519
Another test using the same data, Y)es N)o -->
```

The user is not told to accept or reject the hypothesis; rather, he is given a p-value as shown above. The p-value, or probability level, is an indication of the level of confidence associated with the hypothesis [Ref. 12]. High p-values convey a high confidence in the null

hypothesis; conversely, low p-values reflect a lack of confidence in the validity of the hypothesis. A p-value of .05, for example, indicates that if the hypothesis is indeed true, there is only one chance in twenty that the data used in the test is consistent with the hypothesis. Upon completion of the test, the user has the option to perform another test with the same data or return to the menu for the module in which tests are currently being performed.

V. CONCLUSIONS

The decade of the 1980's promises to be particularly bright in terms of affording the operations research analyst easy access to computing power. Microcomputers now available and their more capable descendants will doubtless play an important role in securing this access. However, equally as important are the software packages which will accompany these computers.

The software package described in this paper provides the analyst with a useful set of statistical tools which can be used on one of the most popular, current microcomputers, the Apple. The five major modules contained in the package (confidence intervals for single and bivariate populations, hypothesis testing for one and two parameters, and the distributions and quantiles) are designed to be easy for the analyst to use and to cushion, as much as possible, potential user mistakes. The algorithms used throughout the program were chosen on the basis of being compatible with the microcomputer with respect to size and computing precision and providing the best combination of speed and accuracy.

Pascal, the programming language used, offers not only the advantages of portability and increased computational speed, but also flexibility. Pascal is flexible in that large complex programs are programmed in modular segments which are then combined into the overall program. It follows that programs developed in this way are easily enhanced by the addition of new modules. Such is the case with this statistical package which could be significantly enhanced by the addition of a regression and an analysis of variance module.

This package and those to follow which are compatible with current and future microcomputers can have a significant impact in the analyst community in two key areas. First, the analyst can be better educated. Simply alleviating the tedium which accompanies the application of many statistical procedures will give fledgling analysts the opportunity to work more problems and be exposed to a greater variety of situations in the school environment. Perhaps of equal importance, the educational environment can provide the opportunity to accustom the analyst to the capabilities that can and should be available for his use in a working environment. Second, by expanding computing power into areas which were not privileged to have it before, the educational process has a better opportunity to continue. Today, it is reasonable to assume that the professional growth of many analysts is stifled from a lack of computing machinery with which to attack his problems.

Taking full advantage of the microcomputer's hardware capabilities requires efficient compatible software. In specialized areas such as operations research, the analysts themselves must logically provide the bulk of the effort in software development. The statistical package which is the subject of this thesis effort scarcely begins to provide the full complement of tools which the analyst requires. If the operations research community is to take advantage in a timely manner of the new opportunities afforded by microcomputers, effort must continue now in software development.

APPENDIX A

```
FUNCTION Z ;

{*****************************************}
{*                                       *}
{*                                       *}
{*          NORMAL DISTRIBUTION          *}
{*       SOME COMMON BASIC PROGRAMS      *}
{*             3RD ED., P. 128           *}
{*                                       *}
{*                                       *}
{*****************************************}

CONST
  C1 = 0.4361836;
  C2 = -0.1201676;
  C3 = 0.937298;
  C4 = 0.33267;
  C5 = 2.5066283;

VAR
  XX,XT,R,T : REAL;

BEGIN
  IF STDEV <= 0.0 THEN
  BEGIN
    ERROR;
    EXIT (Z);
  END
  ELSE
  BEGIN
    XT := X;
    X := ABS((X-MEAN)/STDEV);
    XX := X*X;
    R := EXP(-XX/2.0)/C5;
    X := 1.0/(1.0 + C4*ABS(X));
    T := 0.5 -R*(C1*X +C2*X*X +C3*X*X*X);
    IF XT < MEAN THEN
      Z := 0.5 - T
    ELSE
      Z := T + 0.5;
  END;
END; (* Z *)
```

```
FUNCTION INVZ ;

{**************************************}
{*                                    *}
{*                                    *}
{*           NORMAL QUANTILES         *}
{*   HANDBOOK OF MATHEMATICAL FUNCTIONS *}
{*                P. 933              *}
{*                                    *}
{*                                    *}
{**************************************}

CONST
  C1 = 2.515517;
  C2 = 0.802853;
  C3 = 0.010328;
  D1 = 1.432788;
  D2 = 0.189269;
  D3 = 0.001308;

VAR
  PT,T,NUM,DEN : REAL;

BEGIN
  IF (P >= 1.0) OR (P <= 0.0) THEN
  BEGIN
    ERROR;
    EXIT (INVZ);
  END
  ELSE IF P > 0.5 THEN
    PT := 1.0 - P
  ELSE
    PT := P;
  T := SQRT(LN(1.0/(PT*PT)));
  NUM := C1 + C2*T + C3*T*T;
  DEN := 1.0 + D1*T + D2*T*T + D3*T*T*T;
  IF P > 0.5 THEN
    INVZ := T - NUM/DEN
  ELSE
    INVZ := -(T - NUM/DEN);
END; (* INVZ *)
```

```
FUNCTION T ;

(*****************************************)
(*                                       *)
(*                                       *)
(*          T DISTRIBUTION               *)
(*              CACM                     *)
(*           ALGORITHM 344               *)
(*                                       *)
(*                                       *)
(*****************************************)

CONST C1 = 0.63661977;

VAR
  I,N : INTEGER;
  ANS,D2,F1,F2,T1,T2,XT : REAL;

BEGIN
  IF DF < 1 THEN
  BEGIN
    ERROR;
    EXIT (T);
  END
  ELSE
  BEGIN
    XT := X;
    X := ABS(X);
    T1 := X/SQRT(DF);
    T2 := 1.0/(1.0+T1*T1);
    IF ODD (DF) THEN
    BEGIN
      ANS := 1.0-C1*ATAN(T1);
      IF DF <> 1 THEN
      BEGIN
        D2 := C1*T1*T2;
        ANS := ANS - D2;
      END;
      IF DF <> 3 THEN
      BEGIN
        F1 := 0.0;
        N := (DF-2) DIV 2;
        FOR I := 1 TO N DO
        BEGIN
          F2 := 2.0*I-F1;
          D2 := D2*T2*F2/(F2+1);
          ANS := ANS-D2;
        END;
      END;
    END
    ELSE
    BEGIN
      D2 := T1*SQRT(T2);
      ANS := 1.0-D2;
      IF D2 <> 2 THEN
      BEGIN
        F1 := 1.0;
        N := (DF-2) DIV 2;
        FOR I := 1 TO N DO
        BEGIN
          F2 := 2.0*I-F1;
          D2 := D2*T2*F2/(F2+1);
          ANS := ANS-D2;
        END; (* END FOR *)
      END;
    END; (* END ELSE *)
    IF ANS < 0.0 THEN
```

```
      ANS := 0.0;
    IF XT < 0.0 THEN
      T := ANS/2.0
    ELSE
      T := 1.0 - ANS/2.0;
  END;
END; (* T *)
```

```
FUNCTION INVT ;

{*****************************************}
{*                                       *}
{*                                       *}
{*          T QUANTILES                  *}
{*             CACM                      *}
{*         ALGORITHM 396                 *}
{*                                       *}
{*                                       *}
{*****************************************}

CONST
  HALFPI = 1.570796327;

VAR
  PT,DEN,A,B,C,D,X,Y : REAL;

BEGIN
  PT := P;
  IF (DF < 1) OR (P >= 1.0) OR (P <= 0.0) THEN
  BEGIN
    ERROR;
    EXIT (INVT);
  END
  ELSE IF P > 0.5 THEN
    P := 2.0*(1.0 - P)
  ELSE
    P := 2.0*P;
  IF DF = 2 THEN
    INVT := SQRT(2.0/(P*(2.0 - P))-2.0)
  ELSE
  BEGIN
    IF DF = 1 THEN
    BEGIN
      P := P*HALFPI;
      INVT := COS(P)/SIN(P);
    END
    ELSE
    BEGIN
      A := 1.0/(DF - 0.5);
      B := 48.0/(A*A);
      C := ((20700*A/B - 98.0)*A - 16.0)*A + 96.36;
      D := ((94.5/(B + C) - 3.0)/B + 1.0)*SQRT(A*HALFPI)*DF;
      X := D*P;
      Y := XPN(X,(2.0/DF));
      IF Y > 0.05 + A THEN
      BEGIN
        X := INVZ(P*0.5);
        Y := X*X;
        IF DF < 5 THEN
          C := C + 0.3*(DF - 4.5)*(X + 0.6);
        C := (((0.05*D*X - 5.0)*X - 7.0)*X - 2.0)*X + B + C;
        Y := (((((0.4*Y + 6.3)*Y + 36.0)*Y + 94.5)/C-Y-3.0)/
        B+1.0)*X;
        Y := A*(Y*Y);
        IF Y >0.002 THEN
          Y := EXP(Y) - 1.0
        ELSE
          Y := 0.5*(Y*Y) + Y;
      END
      ELSE
        Y := ((1.0/(((DF+6.0)/(DF*Y)-0.089*D-0.822)*
             (DF+2.0)*3.0)+0.5/(DF+4.0))*Y-1.0)*
             (DF+1.0)/(DF+2.0)+1.0/Y;
    IF PT >= 0.5 THEN
      INVT := SQRT(DF*Y)
```

```
    ELSE
      INVT := - SQRT(DF*Y);
    END;
  END;
END; (* INVT *)
```

```
FUNCTION CHISQ ;
{*****************************************}
{*                                       *}
{*                                       *}
{*        CHI-SQUARE DISTRIBUTION        *}
{*      SOME COMMON BASIC PROGRAMS       *}
{*           (DF <= 40)  P. 130          *}
{*  HANDBOOK OF MATHEMATICAL FUNCTIONS   *}
{*           (DF > 40)  P. 941           *}
{*                                       *}
{*                                       *}
{*****************************************}

VAR
  I : INTEGER;
  Y,POWER,TEMP,NUM,DEN,J,L,M : REAL;

BEGIN
  IF (DF < 1) OR (X <= 0.0) THEN
  BEGIN
    ERROR;
    EXIT (CHISQ);
  END
  ELSE IF DF > 40 THEN
  BEGIN
    Y := ((XPN(X/DF,1.0/3.0)-1.0)+2.0/(9.0*DF))/SQRT(2.0);
    Y := Y*3.0*SQRT(DF);
    IF Y < -4.3 THEN
      CHISQ := 1.0
    ELSE IF Y > 4.3 THEN
      CHISQ := 0.0
    ELSE CHISQ := Z(Y,0.0,1.0);
  END
  ELSE
  BEGIN
    DEN := 1.0;
    TEMP := DF;
    REPEAT
      DEN := DEN * TEMP;
      TEMP := TEMP - 2.0;
    UNTIL TEMP < 2.0;
    POWER := (DF + 1) DIV 2;
    NUM := XPN(X,POWER)*EXP(-X/2.0)/DEN;
    IF ODD (DF) THEN
      J := SQRT(2.0/X/3.1415926)
    ELSE
      J := 1.0;
    L := 1.0;
    M := 1.0;
    REPEAT
      DF := DF + 2;
      M := M*X/DF;
      L := L+M;
    UNTIL M < 0.0000001;
    L := L-1;
    CHISQ := J*NUM*L;
  END;
END; (* CHISQ *)
```

```
FUNCTION INVCHI ;

{*****************************************}
{*                                       *}
{*                                       *}
{*         CHI-SQUARE QUANTILES          *}
{*               CACM                    *}
{*           ALGORITHM 451               *}
{*                                       *}
{*                                       *}
{*****************************************}

CONST
 C1  = 1.565326E-3;   C2  = 1.060438E-3;   C3  = -6.959356E-3;
 C4  = -1.323293E-2;  C5  = 2.277679E-2;   C6  = -8.986007E-3;
 C7  = -1.51390E-2;   C8  = 2.53001E-3;    C9  = -1.450117E-3;
 C10 = 5.169654E-3;   C11 = -1.153751E-2;  C12 = 1.126186E-2;
 C13 = 2.607083E-2;   C14 = -2.237368E-1;  C15 = 9.780499E-5;
 C16 = -8.426812E-4;  C17 = 3.125580E-3;   C18 = -8.553069E-3;
 C19 = 1.348028E-4;   C20 = 4.713941E-1;   C21 = 1.0000886;
 A1  = 1.264461E-2;   A2  = -1.425296E-2;  A3  = 1.400483E-2;
 A4  = -5.886090E-3;  A5  = -1.091121E-2;  A6  = -2.304527E-2;
 A7  = 3.135411E-3;   A8  = -2.728484E-4;  A9  = -9.699681E-3;
 A10 = 1.316872E-2;   A11 = 2.618891E-2;   A12 = -2.222222E-1;
 A13 = 5.406667E-5;   A14 = 3.483789E-5;   A15 = -7.274761E-4;
 A16 = 3.292181E-3;   A17 = -8.729713E-3;  A18 = 4.714045E-1;
 A19 = 1.0;

VAR
  F,F1,F2,TEMP : REAL;

BEGIN
  IF (DF < 1) OR (P <= 0.0) OR (P >= 1.0) THEN
  BEGIN
    ERROR;
    EXIT (INVCHI);
  END
  ELSE
  BEGIN
    P := 1.0 - P;
    IF DF = 1 THEN
    BEGIN
      TEMP := INVZ(0.5*P);
      INVCHI := TEMP*TEMP;
    END
    ELSE IF DF = 2 THEN
      INVCHI := -2.0*LN(P)
    ELSE
    BEGIN
      F := DF;
      F1 := 1.0/F;
      TEMP := INVZ(1.0-P);
      F2 := SQRT(F1)*TEMP;
      IF DF < ( 2 + TRUNC(4.0*ABS(TEMP))) THEN
        TEMP := ((((((C1*F2+C2)*F2+C3)*F2+C4)*F2
                  +C5)*F2+C6)*F2+C7)*F1+(((((C8+C9*F2)*F2
                  +C10)*F2+C11)*F2+C12)*F2+C13)*F2+C14))*F1+
                  ((((C15*F2+C16)*F2+C17)*F2+C18)*F2
                  +C19)*F2+C20)*F2+C21
      ELSE
        TEMP := (((A1+A2*F2)*F1+(((A3+A4*F2)*F2
                  +A5)*F2+A6))*F1+((((A7+A8*F2)*F2+A9)*F2
                  +A10)*F2+A11)*F2+A12))*F1+(((((A13*F2
                  +A14)*F2+A15)*F2+A16)*F2+A17)*F2*F2
                  +A18)*F2+A19;
      INVCHI := TEMP*TEMP*TEMP*F;
    END;
```

```
    END;
END;  (* INVCHI *)
```

```
FUNCTION F1;

(*****************************************)
(*                                       *)
(*                                       *)
(*            F DISTRIBUTION             *)
(*      SMALL DEGREES OF FREEDOM         *)
(*        STATISTICAL COMPUTING          *)
(*              P. 114                   *)
(*                                       *)
(*                                       *)
(*****************************************)

CONST PI = 3.14159265;

VAR A,A1,AN1,AN2,ART,D1,D2,D3,R,S1,S2,DEL,XM,XK,T,
    TEM,C : REAL;
    I,M,N : INTEGER;

BEGIN
  AN1 := DF1;
  AN2 := DF2;
  A := DF1*X/(DF1*X+DF2);
  A1 := 1 - A;
  IF A1 <= 0.0 THEN
    A1 := 1.0E-37;
  D1 := AN1*0.5;
  D2 := AN2*0.5;
  D3 := D1 + D2 - 1.0;
  R := 0.0;
  S1 := 0.0;
  S2 := 0.0;
  DEL := 1.0;
  XM := 1.0;
  XK := 1.0;
  C := 0.25;
  N := DF2;
  WHILE  C < 0.875 DO
  BEGIN
    M := TRUNC(D2);
    M := 2 * M;
    IF M = N THEN
    BEGIN
      N := TRUNC(D2)-1;
      IF N > 0 THEN
      BEGIN
        FOR I := 1 TO N DO
        BEGIN
          S1 := DEL+S1*R;
          D2 := D2 - 1.0;
          D3 := D3 - 1.0;
          TEM := A1/D2;
          R := D3*TEM;
          S2 := (R+TEM)*S2;
        END;
      END;
      S1 := DEL+S1*R;
      DEL := 0.0;
      T := -1.0;
      D3 := -1.0;
      S2 := A*S2;
      C := C + 0.5;
    END
    ELSE  (* DEGREES OF FREEDOM ODD *)
    BEGIN
      N := TRUNC(D2);
      IF N <> 0 THEN
```

```
      BEGIN
        FOR I := 1 TO N DO
        BEGIN
          S1 := DEL+S1*R;
          D2 := D2 - 1.0;
          D3 := D3 - 1.0;
          TEM := A1/D2;
          R := D3*TEM;
          S2 := (R+TEM)*S2;
        END; (* FOR *)
      END;
      S1 := XK*S1;
      S2 := XK*S2;
      ART := SQRT(A1);
      XM := XM*ART;
      T := (XM-ART)/A1;
      D3 := -0.5;
      XK := 2.0/PI;
      C := C*2.0;
    END;
    IF C <= 0.875 THEN
    BEGIN
      D2 := D1;
      D3 := D2 + D3;
      S2:= S1;
      S1 := 0.0;
      A1 := A;
      IF A1 <= 0.0 THEN
        A1 := 1.0E-37;
      N := DF1;
    END;
  END; (* WHILE *)
  IF C < 1.125 THEN
    DEL := 4.0/PI*ATAN(T);
  F1 := XM*(S2 - S1) - DEL;
END; (* F1 *)
```

```
FUNCTION F2;

(*****************************************************)
(*                                                   *)
(*                                                   *)
(*                  F DISTRIBUTION                   *)
(*             LARGE DEGREES OF FREEDOM              *)
(*   EDUCATIONAL AND PSYCHOLOGICAL MEASUREMENT       *)
(*             V. 25, NO.3, P. 877-879               *)
(*                                                   *)
(*                                                   *)
(*****************************************************)

CONST
  C1 = 0.196854;C2 = 0.115194;C3 = 0.000344;C4 = 0.019527;

VAR
  S,T,Z,Y,XX,YY,J,K : REAL;

BEGIN
  IF X > 1 THEN
  BEGIN
    S := DF1;
    T := DF2;
    Z := X;
  END
  ELSE
  BEGIN
    S := DF2;
    T := DF1;
    Z := 1.0/X;
  END;
  J := 2.0/9.0/S;
  K := 2.0/9.0/T;
  Y := ABS((1.0-K)*XPN(Z,(1.0/3.0))-1.0+J)/
  SQRT(K*XPN(Z,(2.0/3.0))+J);
  YY := 1.0 + Y*(C1+Y*(C2+Y*(C3+Y*C4)));
  XX := 0.5/XPN(YY,4.0);
  IF X >= 1.0 THEN
    F2 := 1.0 - XX
  ELSE
    F2 := XX;
END; (* F2 *)

FUNCTION F;

BEGIN
  IF (DF1 < 1) OR (DF2 < 1) OR (X < 0.0) THEN
  BEGIN
    ERROR;
    EXIT (F);
  END;
  IF ((DF1 < 100) AND (DF2 < 100)) OR
  (DF1 < 20) OR (DF2 < 20) THEN
    F := F1 (X,DF1,DF2)
  ELSE
    F := F2 (X,DF1,DF2);
END; (* F *)
```

```
FUNCTION INVF ;

(*****************************************)
(*                                       *)
(*                                       *)
(*            F QUANTILES                *)
(*        BISECTION SEARCH FOR           *)
(*      SMALL DEGREES OF FREEDOM         *)
(*      LARGE DEGREES OF FREEDOM         *)
(*  HANDBOOK OF MATHEMATICAL FUNCTIONS   *)
(*              P. 947                   *)
(*                                       *)
(*                                       *)
(*****************************************)

CONST EPS = 0.005;

VAR
  TEMP,PT,ENDR,ENDL,MIDPT,STEP,W,H,T,Y,Z : REAL;

BEGIN
  IF (DF1 < 1) OR (DF2 < 1) OR (P <= 0.0) OR (P >= 1.0) THEN
  BEGIN
    ERROR;
    EXIT (INVF);
  END
  ELSE IF (DF1 = 1) OR (DF2 = 1) THEN
  BEGIN
    IF DF1 = 1 THEN
      INVF := SQR(INVT((0.5*(1.0+P)),DF2))
    ELSE
      INVF := 1.0/SQR(INVT((1.0-P/2.0),DF1));
  END
  ELSE
  BEGIN
    Z := INVZ(P);
    Y := (Z*Z - 3.0)/6.0;
    H := 2.0 * 1.0/(1.0/(DF2-1.0) + 1.0/(DF1-1.0));
    W := (Z*SQRT(H+Y))/H - (1.0/(DF1-1.0)-1.0/(DF2-1.0))*
         (Y + 5.0/6.0 -2.0/(3.0*H));
    T := EXP(2.0*W);
    IF (DF1 <= 15) OR (DF2 <= 15) THEN
    BEGIN
      STEP := 0.1*T;
      ENDR := T;
      REPEAT
        TEMP := ENDR;
        ENDL := ENDR - STEP;
        IF ENDL <= 0.0 THEN
          ENDL := 0.0;
        ENDR := ENDL;
        PT := F1 (ENDL,DF1,DF2);
      UNTIL (PT <= P) OR ((TEMP - ENDL) <= EPS);
      ENDR := TEMP;
      IF ABS(ENDR - ENDL) > EPS THEN
      BEGIN
        WHILE (ENDR - ENDL) > 0.1 DO
        BEGIN
          T := (ENDR + ENDL)/2.0;
          PT := F1 (T,DF1,DF2);
          IF PT < P THEN
            ENDL := T
          ELSE
            ENDR := T;
        END;
      END;
```

```
      INVF := (ENDL + ENDR)/2.0;
    END
    ELSE
      INVF := F;
  END; (* ELSE *)
END; (* INVF *)
```

APPENDIX B

```
PROCEDURE CONVERTSTRING;

{*****************************************}
{*                                       *}
{*     PROCEDURE TO CONVERT A STRING     *}
{*     VARIABLE TO A NUMBER              *}
{*                                       *}
{*****************************************}

LABEL 1;

VAR
  PLACEINT,SIGN,I : INTEGER;
  BFDEC : BOOLEAN;
  ACC,TACC,PLACE : REAL;
  PREVCH,CH : CHAR;

BEGIN
  DATA := CONCAT (DATA,' ');
  PREVCH := ' ';
  BFDEC := TRUE;
  INT := 0;
  RL := 0.0;
  ACC := 0.0;
  PLACE := 1;
  SIGN := 1;
  FOR I := 1 TO LENGTH (DATA) DO
  BEGIN
    CH := DATA[I];
    IF CH = PREVCH THEN
      GOTO 1;
    PREVCH := '/';
    IF CH IN ['0'..'9','.',' ','-',','] THEN
    CASE CH OF
      '0','1','2','3','4','5','6','7','8','9':
        BEGIN
          IF BFDEC THEN
          BEGIN
            RL := RL * PLACE + (ORD (CH)-48) ;
            PLACE := 10;
          END
          ELSE
          BEGIN
            PLACE := PLACE * 0.1;
            RL := RL + PLACE * (ORD(CH)-48)
          END;
        END;
      '.':
        BEGIN
          BFDEC := FALSE;
          PLACE := 1
        END;
      ' ':
        BEGIN
          RL := RL * SIGN;
          IF ABS (RL) < 32767.0 THEN   (* MAX INTEGER *)
            INT := TRUNC (RL)          (* IS 32767    *)
          ELSE
            INT := 1;
        END;
      '-':
```

```
            BEGIN
              SIGN := -1;
              PREVCH := ' ';
            END;
          ',':GOTO 1;
      END; (*CASE*)
    1: END;
END; (* CONVERTSTRING *)
```

LIST OF REFEERENCES

1. Mastrakas, M., A Method for Evaluation of Microcomputers for Tactical Applications, Masters Thesis, Naval Postgraduate School, 1980.

2. Ling, R. F., "General Considerations on the Design of an Interactive System for Data Analysis," Communications of the ACM, v. 23, p. 147-154, March 1980.

3. Muller, M. E., "Aspects of Statistical Computing: What Packages for the 1980's Ought To Do," The American Statistician, v. 34, no. 3, p. 159-158, August 1980.

4. Bowles, K. L., Beginner's Guide for the UCSD Pascal System, p. 121-155, Byte/McGraw-Hill, 1980.

5. Kellner, J., "Pascal Operand Formats," Apple Orchard, v. 1, no. 2, p. 39, 1980.

6. Forsythe, A. B., "Elements of Statistical Computation," Byte, p. 182-184, January 1979.

7. Dixon, W. J. and Massey, F. J., Introduction to Statistical Analysis, 3rd ed., p. 125-140, Osborne/McGraw-Hill, Inc., 1969.

8. Kennedy, W. J. and Gentle, J. E., Statistical Computing, p. 112-116, Dekker, 1980.

9. Johnson, N. L. and Kotz, S., Continuous Univariate Distributions, v. 2, Wiley, 1970.

10. Storer, R., Statistics Programs for the TI-59 Calculator, Masters Thesis, Naval Postgraduate School, 1980.

11. Apple Computer, Inc., Apple Pascal Reference Manual, 1979.

12. Gibbons, J. and Pratt, J. W., "P-Values: Interpretation and Methodology," The American Statistician, v. 29, no. 1, p. 20-25, February 1975.

BIBLIOGRAPHY

Abramowitz, M. and Stegun, I. A., Handbook of Mathematical Functions, p. 927-949, Dover Publications, 1979.

Association of Computing Machinery, "Collected Algorithms from ACM", 1980.

Dixon, W. J. and Massey, F. J., Introduction to Statistical Analysis, 3rd ed., p. 125-140, Osborne/McGraw-Hill, Inc., 1969.

Golden, R. R., Weiss, D. J., and Dawiss, R. V., "An Evaluation of Jaspen's Approximation of the Distribution Functions of the F, T, and Chi-square Statistics," Educational and Psychological Measurement, v. 28, p. 163-165, 1968.

Jaspen, N., "The Calculation of Probabilities Corresponding to Values of Z, T, F, and Chi-square," Educational and Psychological Measurement, v. 25, no. 3, p. 877-879, 1965.

Johnson, N. L. and Kotz, S., Continuous Univariate Distributions, v. 1, Wiley, 1970.

Mood, A. M. and Graybill, F. A., Introduction to the Theory of Statistics, 2nd ed., McGraw-Hill, 1963.

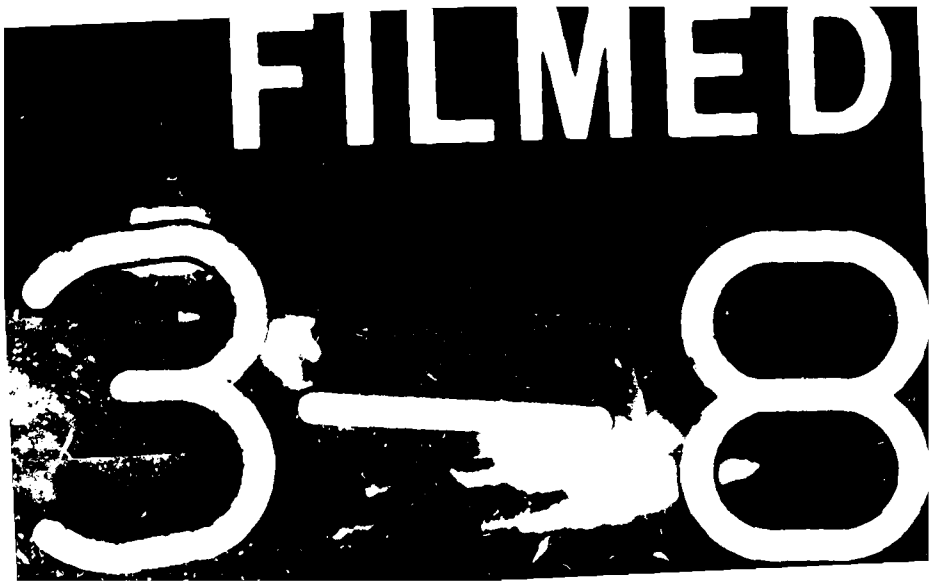
Poole, L. and Borchers, M., Some Common BASIC Programs, 3rd ed., p. 125-140, Osborne/McGraw-Hill, 1979.

Smillie, K. W. and Anstey, T. H., "A Note on the Calculation of Probabilities in an F-Distribution," Communications of the Association of Computing Machinery, v. 7, p. 725, 1964.

Zehna, P. W., Introductory Statistics, p. 191-280, Prindle, Weber and Schmidt, Inc., 1974.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22314	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, California 93940	2
3.	Professor D. R. Barr, Code 55Bn Department of Operations Research Naval Postgraduate School Monterey, California 93940	1
4.	LCDR C. F. Taylor, Code 55Ta Department of Operations Research Naval Postgraduate School Monterey, California 93940	1
5.	MAJ R. N. Duff 9 Revere Road Monterey, California 93940	1
6.	MAJ J. D. Morgeson 908 Virginia Crane, Texas 79731	2


FILMED
3-8